logic analyzer to run and to begin capturing data. Once a breakpoint occurs, the logic analyzer determines if more samples need to be captured after the breakpoint. The EDA tool then directs the logic analyzer to unload the data from sample memory and then displays the data on the computer. The logic analyzer circuit may then run again to capture another sequence of sample values.

Brief Summary Text (29):
In one specific embodiment of the invention, the logic analyzer captures data from specified signal lines continuously in a ring buffer, or similar memory structure that overwrites earlier stored data when full. In this fashion, relevant data is stored continuously before a breakpoint occurs, thus, the stored data may be viewed later by a user who wishes to view signals occurring before the breakpoint. Once the breakpoint occurs, a counter keeps track of how many additional samples of data (if any) need be collected.

Brief Summary Text (31):
The present invention provides many advantages over the prior art. Use of an embedded logic analyzer in a PLD allows debugging of the device in the system in which it is operating and under the actual conditions that might produce a malfunction of the PLD. The technique of the present invention automatically embeds a logic analyzer circuit into a PLD so that an engineer may debug any logic function within the device. The embedded logic analyzer is able to capture any internal signals specified by the engineer; the breakpoint can also include any specified internal signals. Through the use of memory within the embedded logic analyzer and an interface to the computer, any number and depth of signals can be monitored within the device and then transmitted to the computer at a later time for analysis. In one embodiment of the invention, a JTAG port is used to program the embedded logic analyzer and to transmit captured signal information to the computer.

Brief Summary Text (33):
The present invention is applicable to a wide range of hardware devices, and especially to PLDs. A PLD in particular may be implemented using a wide variety of technologies, including SRAM technology and EEPROM technology. PLDs based upon SRAM technology are especially advantageous in that they may have additional embedded memory that can be used by the embedded logic analyzer to capture a large number of, and a greater depth of signals. Furthermore, an embedded logic analyzer that is designed and inserted automatically by an EDA tool means that an engineer does not require an external logic analyzer as a separate piece of equipment. Furthermore, the engineer may use the computer on which he or she is creating a design for the PLD to also control and configure the embedded logic analyzer and to review its results.

Brief Summary Text (34):
In one embodiment of the present invention, a number of pins on the PLD are dedicated interface pins for communication with the user computer. Because these pins are dedicated for the interface, and are known ahead of time, they may be routed to an easily accessible location or port on a circuit board, such that a debugging interface cable may be connected from the user computer to these pins extremely easily. This technique is especially advantageous where pins or contacts of a particular integrated circuit in a package may be difficult or nearly impossible to reach. Because the embedded logic analyzer of the present invention may be configured to monitor any internal or external signals of the PLD, all of these monitored signals are available for analysis through these interface pins. In other words, it is not necessary to physically connect a probe to a particular external pin of interest because any signal within the PLD can be monitored, stored within the memory of the embedded logic analyzer and then later uploaded to the user computer for analysis through these dedicated interface pins.

Detailed Description Text (30):
In step 110, the total number of samples to be captured are specified. In other words, the depth of the sample memory is specified and this, in turn, indicates on how many clock pulses data will be acquired by the logic analyzer. The total number of samples to be captured includes all samples to be captured whether before or after a specified breakpoint. In other words, the specified number indicates a width of a window of data to be captured; the window may encompass the breakpoint, may occur completely before the breakpoint, or may occur completely after the breakpoint.

Detailed Description Text (31):
In one embodiment of the invention, a PLD that includes embedded memory blocks (such as any of the FLEX 10K family of devices available from Altera Corporation) works well for

implementing the present invention. The embedded memory blocks are easily programmed to provide relatively large buffers (as part of the logic analyzer circuit) for the storage of captured information. Embedded memory devices are not, however, necessary for the buffering of information captured. Although devices without embedded memory may be used with the present invention, they do not lend themselves as easily to the creation of relatively large buffers. In devices without embedded memory, buffers may be implemented over multiple cells, using available memory from each cell.

Detailed Description Text (35):
In addition to being able to specify number of samples to be captured in step 110 and number of samples needed prior to the breakpoint in step 115, these values may also be specified after the user's design and logic analyzer have been compiled. In other words, while steps 110 and 115 specify certain values before compilation of the design, these values may also be input to the logic analyzer once the PLD has been programmed or even while the logic analyzer is running. For example, register 310 of FIG. 8 may be set at any with the value Delay [6:0] to indicate the number of samples to be captured after the breakpoint occurs. The total number of samples to be captured is equal to the number of words in sample memory 324. Other values specified before compilation can also be specified after compilation in a similar fashion, for example trigger register 304.

Detailed Description Text (40):
In step 122 the user through the EDA tool requests the embedded logic analyzer to begin running with an appropriate command. Once the logic analyzer begins to run, in step 124 it begins to continuously capture data from the signals that have been specified to be monitored. Preferably, the user then manipulates the system to duplicate previous malfunctions that the user wishes to analyze The captured data is stored within memory of the PLD, and is preferably stored within dedicated memory with in the embedded logic analyzer itself. Step 126 determines weth point has occurred. In other words, the logic analyzer determines whether the state of the signals specified to be monitored are equivalent to the e breakpoint that the user has specified. If not, then the logic analyzer continues to capture data. If so, step 128 determines whether more samples should be captured and stored after the breakpoint. Step 128 may be implemented by comparing the total number of samples desired with the number of samples that have already been stored prior to the breakpoint. If more samples are to be stored, then in step 130 the logic analyzer continues to store the desired number of samples after the breakpoint.

Detailed Description Text (41):
Once the total number of samples desired by the user have been captured and stored, the stored data is dumped from the sample memory of the logic analyzer to the user's computer in step 132. Preferably the logic analyzer uploads this stored information over the interface cable to the user's computer. In step 134 the user is able to graphically view these signals received from the logic analyzer. In one embodiment, the signals are presented in a waveform view annotated with the names of the signals. Thus, by viewing these signals of interest on a computer, a user is able to efficiently debug a hardware device in much the same way as if an external logic analyzer had been able to be connected to these signals.

Detailed Description Text (46):
The actual gate level representation of a particular logic analyzer circuit will depend upon the particular device in which the logic analyzer will be embedded. By way of example, the hardware device in which to embed the logic analyzer may include any of the PLD devices available from Altera Corporation. In particular, any of the FLEX 10K, FLEX 8000, MAX 9000, or MAX 7000 devices work well. Each of these particular devices may have different features that would affect how a gate level representation for a logic analyzer is produced. For example, for a FLEX 10K device with relatively large embedded memory sections, the is embedded memory is particularly well suited for implementing a large FIFO (first in first out) memory for the logic analyzer. For a device such as the FLEX 8000 without embedded memory, the memory elements (such as SRAM flip-flops) of logic cells may be used for the memory of the logic analyzer but the FIFO buffer may have to be divided over multiple cells if the memory in a single cell is not sufficiently large to accommodate the e buffer. Similarly, a device based upon EEPROM technology may also use one or more of its logic cells for the logic analyzer's buffer. A device having large embedded memory works particularly well with the present invention because of the larger capacity for signal storage. Thus, step 206 produces a representation for a logic analyzer circuit that is to be connected to the user's design.

Detailed Description Text (58):
Signal NextReq 284 is received from computer system 18 and allows retrieval of stored sample data a sample at a time, and indicates that the next sample should be uploaded to computer system 18. Signal StopReq 285 is received from computer system 18 and directs the logic analyzer to enter its stop state and to stop capturing signal samples. Signal RunReq 286 is received from computer system 18 and directs the logic analyzer to begin running and capturing sample data. Signal DoneDump 287 directs the logic analyzer to discontinue dumping data from its memory to the computer system and to enter a stop state. This signal may be received from within the logic analyzer or from the user. Signal Clock 288 is the system clock signal specified in step 112. Signal Clear 289 is a reset signal that clears control state machine 302, sample memory 324 and counter 314.

Detailed Description Text (59):
Signals DataOut 290 are the output signals from sample memory 324 that transfer captured signals a word at a time from logic analyzer 260 to computer system 18 via interface 264. Signal NumSamples 291 indicates the number of valid samples captured by logic analyzer 260. Because the actual number of valid samples captured by the logic analyzer may be less than the total number of samples requested by the user, this signal assists the user in determining which are the valid samples stored in memory. For example, the user may desire to capture a total of 128 samples but does not wish to store any samples after a breakpoint occurs. If a breakpoint occurs after only sixty-four samples have been captured, the signal NumSamples 291 will have a value of sixty-four, thus indicating that only sixty-four samples stored in the logic analyzer are valid samples. Any samples stored beyond the sixty-four will not be valid samples; they may have been present from an earlier data capture. Signal Triggered 292 is an output signal for the user that indicates that a breakpoint has occurred. Signal Run 293 indicates to the user that signal RunReq 286 has been received and that the logic analyzer is running and capturing data.

Detailed Description Text (60):
FIG. 8 illustrates embedded logic analyzer 260 according to one embodiment of the present invention. A logic analyzer to be embedded within a PLD may be implemented in a wide variety of manners depending upon the type of PLD, signal type and number to be monitored, depth of data desired, memory available, control signals from the user's computer and preferences of the designing engineer, etc. By way of example, logic analyzer 260 is one particular example of how such a logic analyzer may be implemented. The embedded logic analyzer is controlled by the user from a computer external to the PLD and operates to capture any of a variety of internal signals that the user wishes. In this embodiment of the invention, logic analyzer 260 includes control state machine 302, trigger register 304, trigger comparator 306, registers 308 and 310, counters 312-316, comparators 320, 322 and sample memory 324.

Detailed Description Text (62):
Control state machine 302 may be any suitable control structure for controlling the embedded logic analyzer and is described in greater detail in FIG. 9. Preferably, state machine 302 is implemented in programmable logic using any of a variety of look-up tables, embedded memory blocks, ROM registers, etc. Input signal DelayDone is received from comparator 320 and indicates that the total number of samples requested by the user have been captured. Signals NextReq, StopReq, RunReq, DoneDump, Clock and Clear have been described above. Input signal Breakpoint to state machine 302 is received from trigger comparator 306 via register 308.

Detailed Description Text (64):
Output signal Stopped is active when state machine 302 is in its stop state. This signal resets counters 312 and 316 and prepares the logic analyzer to begin running anew. It also permits data from register 310 to be loaded into counter 314. Output signal Next when active enables counter 312 to increment an address in sample memory 324. Addresses are incremented while the logic analyzer is running and capturing data and while sample memory 324 is stepping though its addresses and dumping sample data to computer system 18. Output signal Triggered is used to enable counter 314. Signal Run is combined in gate 340 with signal PrevDataFull to enable counter 316.

Detailed Description Text (66):
Counter 312 increments addresses for sample memory 324 during sampling and capturing data, and increments addresses during reading data out to computer system 18. Counter 314 is a down counter that creates a delay from an observed breakpoint in order to allow the logic analyzer to continue capturing data until the last sample desired by the user has been stored. Counter 314 is loaded from register 310 with signal Delay

which indicates the number of samples past the breakpoint that should be stored. Once counter 314 counts down and reaches a value of zero, comparator 320 performs a successful comparison with a hardwired zero value and asserts signal DelayDone. Signal DelayDone when asserted instructs control state machine 302 to move from a run state to a data dump state.

Detailed Description Text (67):
Counter 316 counts the number of valid samples that have been stored in sample memory 324. As discussed above, a situation may occur in which the number of valid samples captured is less than the originally specified total number of samples desired by the user. In this illustrative example, sample memory 324 has a capacity of 128 words. Thus, when counter 316 reaches the value 128, comparator 322 performs a successful comparison with a hardwired value 128. Accordingly, output signal PrevDataFull is asserted and its inverse is input to gate 340. The inverse of an asserted signal PrevDataFull disables counter 316, thus indicating the total number of valid samples stored in sample memory 324. Counter 316 also continuously outputs signal NumSamples that indicates the total number of valid samples that have been captured to that point in time.

Detailed Description Text (68):
Sample memory 324 is memory within PLD 16 that may be implemented in any suitable fashion. By way of example, memory 324 may be implemented using sets of registers or using embedded memory blocks within the PLD. In one specific embodiment of the invention, embedded SRAM memory is used to implement memory 324. Of course, memory 324 may be implemented in many types of PLDs that do not contain extra embedded memory. Preferably, sample memory 324 is implemented as a ring buffer such that incoming sample data is continuously stored in memory 324 when the logic analyzer is running. As memory 324 fills, it wraps around to its beginning and the oldest data stored is overwritten by new incoming data. For illustrative purposes, sample memory 324 is shown implemented using RAM memory of 128 words of 16 bits each. Of course, any of a variety of sizes of memory may be used. The width of the memory may be increased to capture more data signals and its depth may be increased to store a greater history of signals.

Detailed Description Text (73):
While in the DataDump state, all output signals are unasserted. In this state, data is dumped from sample memory 324 to computer system 18. When the signal DoneDump is asserted, the logic analyzer has finished uploading data to computer system 18 and the state machine moves from the DataDump state to the Stop state. The state machine remains in the DataDump state while signals NextReq and DoneDump are both unasserted. As described earlier, it is possible for computer system 18 to request an upload of a word at a time from sample memory 324. In this situation, computer system 18 asserts input signal NextReq causing the state machine to enter the Next state.

Detailed Description Text (112):
FIG. 18 illustrates a computer system 900 in accordance with an embodiment of the present invention. Computer system 900 includes any number of processors 902 (also referred to as central processing units, or CPUs) that are coupled to storage devices including primary storage 906 (such as random access memory, or RAM) and primary storage 904 (such as a read only memory, or ROM). As is well known in the art, primary storage 904 acts to transfer data and instructions uni-directionally to the CPU and primary storage 906 is used typically to transfer data and instructions in a bi-directional manner. Both of these primary storage devices may include any suitable of the computer-readable media described below. A mass storage device 908 is also coupled bi-directionally to CPU 902 and provides additional data storage capacity and may also include any of the computer-readable media described below. Mass storage device 908 may be used to store programs, data and the like and is typically a secondary storage medium (such as a hard disk) that is slower than primary storage. It will be appreciated that the information retained within mass storage device 908, may, in appropriate cases, be incorporated in standard fashion as part of primary storage 906 as virtual memory. A specific mass storage device such as a CD-ROM 914 passes data uni-directionally to the CPU.

CLAIMS:

1. An electronic design automation (EDA) software tool used to program a programmable logic device (PLD), said EDA tool comprising: a user electronic design intended for said PLD; an indication of internal signals of said electronic design to monitor; an indication of a breakpoint that specifies the state of at least one signal within said electronic design; an indication of a number of samples of said internal signals to be

captured before said breakpoint; a logic analyzer design arranged to store said samples from said electronic design in a memory of said logic analyzer such that said samples are stored before the occurrence of said breakpoint; and a compiler arranged to compile said user electronic design and said logic analyzer design into an output file, whereby said PLD is programmed with said output file.

6. A method for receiving sample data from a logic analyzer embedded within a programmable logic device (PLD), said method comprising: establishing communication with a logic analyzer embedded within a programmable logic device (PLD); specifying a breakpoint indicative of the state of at least one signal within said PLD; indicating to said logic analyzer to continuously store internal signals of said PLD in a memory of said logic analyzer such that said internal signals are stored before the occurrence of said breakpoint; and receiving said stored internal signals from said logic analyzer, said stored signals representing at least signals stored before said breakpoint, whereby said stored internal signals may be viewed on a user computer.

9. A programmable logic device (PLD) comprising: PLD circuitry representing one iteration of an electronic design in a design process to create a final PLD; logic analyzer circuitry integrated within said PLD circuitry such that a portion of said PLD circuitry is connected to said logic analyzer circuitry; a state machine within said logic analyzer circuitry for initiating the capture of signals from said PLD circuitry before a breakpoint occurs; and sample memory circuitry within said logic analyzer circuitry arranged to capture said signals before said breakpoint occurs, whereby said sample memory circuitry is available to present said captured signals for analysis after said capture.

10. A programmable logic device (PLD) as recited in claim 9 wherein said logic analyzer circuitry continues the capture of signals from said PLD circuitry after said breakpoint occurs, said sample memory circuitry being further arranged to capture said signals after said breakpoint occurs, whereby said sample memory circuitry is available to present said captured signals for analysis after said capture.

11. A programmable logic device as recited in claim 9 wherein said sample memory circuitry includes a ring buffer that overwrites earlier stored signals when full, whereby signals stored before said breakpoint are made available for later analysis.